DEPARTAMENTO DE
ELECTRÓNICA
UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

ELO323 - REDES DE COMPUTADORES II
PROYECTO

MEDICIÓN POTENCIA EN TRANSMISIÓN INALÁMBRICA ENTRE DOS TMOTE SKY

Profesor
Agustín González

Integrantes
Camilo Barra
Roberto Cifuentes
Oscar Silva

Resumen

Las redes de sensores son tema de mucho estudio en la actualidad en muchas universidades y que poseen un creciente número de aplicaciones en áreas como la medicina, domótica,industria, sensado ambiental, etc. Es por estos motivos que se han utilizado redes de sensores para el proyecto a presentar. El proyecto consta de la comprobación de la capacidad de comunicarse de forma inalámbrica de los nodos, así como la medición de la potencia recibida. Para poder realizar esto se envían paquetes de mensajes entre dos Tmote, y se procede a extraer las mediciones de RSSI(Received signal strength indication), las cuales las entrega la antena incorporada en los sensores, todo esto en el ambiente dado por el TyniOS, sistema operativo creado para las redes de sensores, así como el lenguaje de programación NesC. Se pudo apreciar al realizar los experimentos que la intensidad de la señal recibida decrece con la distancia.

Introducción

Redes de sensores

Las redes de sensores permiten monitorizar y controlar eventos que ocurren en el entorno, pudiendo actuar sobre este o informar a una estación. Con una cierta cantidad de sensores distribuidos se puede recolectar la información obtenida del medio y enviarla a través de los demás sensores (nodos) a un nodo gateway, él cual puede enviar la información a través de internet para luego ser procesada.

Tmote Sky

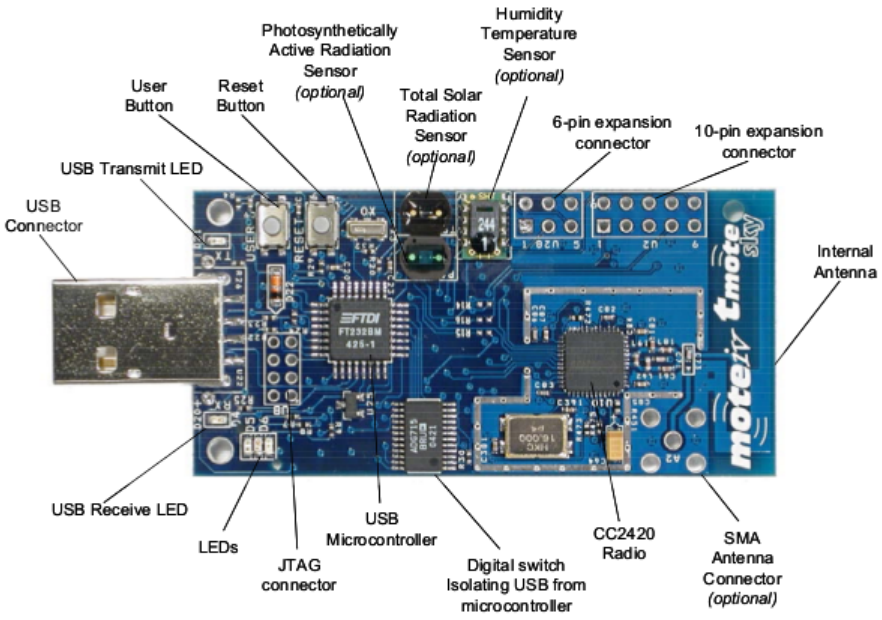
Dentro de la variedad de dispositivos que se dedican a formar redes de sensores inalámbricas(WSN: wireless sensor network), existe **Tmote Sky**, él cual cuenta con una serie de sensores que permiten medir temperatura, humedad relativa e luminancia. Son dispositivos de bajo consumo, con una alta velocidad de transmisión y alta tolerancia a fallos para un fácil desarrollo orientados en redes de sensores.

Tmote fueron desarrollados por la University of California,Berkeley, EE.UU utilizan el sistema operativo TinyOS , un sistema de código abierto, orientado a componentes. Los componentes de este sistema están desarrollados en el lenguaje nesC, al igual que las aplicaciones que se distribuyen y desarrollan para TinyOS.

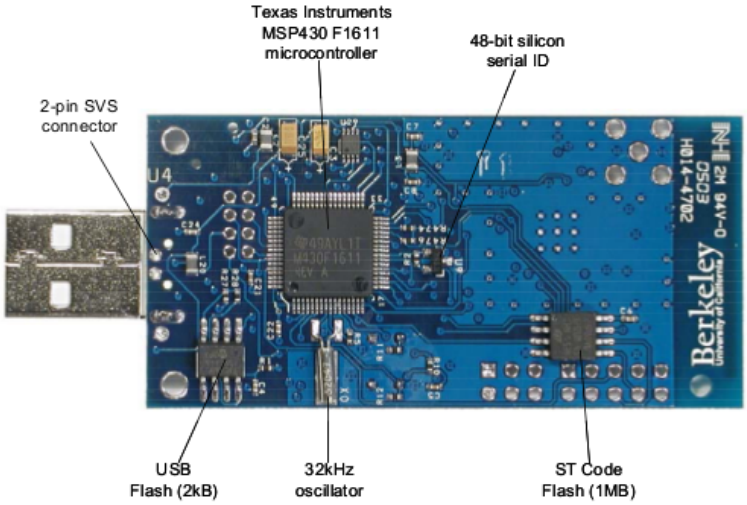
Fig. 1: Dispositivo Tmote Sky

Unidad de procesamiento		**Interfaz**	
MCU	MSP430F1611	**Digital**	6 GPIO
RAM	10k RAM	**Analógica**	6 ADC
ROM	48K Flash	**Serial**	I^2C,SPI, USB virtual serial com port
Velocidad	8 MHz	**Sensor**	Humedad, luz y temperatura
SO	TinyOS	**Intraestrucctura**	Ethernet
Radio Frecuencia		**Otros**	
Trasnductor	Chipcon CC2420	**Voltaje**	2.1 - 3.6 V
Banda	2400 – 2483.5 MHz	**Consumo reposo**	5.1 uA
Tasa Bits	250 kbps	**Rx/Tx**	19.5 mA / 21.8 mA
Modulación	O-QPSK	**Precio**	€90
Rango	125 m Exterior, 50 m interior	**Gateway/mesh**	Tmote Connect
Protocolo	802.15.4	**Licencia**	BSD (Sistema Operativo)
Seguridad	Los que posee el protocolo 802.15.4	**Extras**	Ninguno

Tabla 1: Especificaciones Tmote Sky



(a) Vista frontal



(b) Vista trasera

Fig. 2: Componentes Tmote Sky

NesC

NesC es un lenguaje de programación basado en componentes, y eventos. Es utilizado para construir aplicaciones para el sistema operativo TinyOS. Este SO está diseñado para redes de sensores que poseen pocos recursos(ej: 8K bytes de memoria programable, 512 bytes de RAM). NesC está construido como una extensión del lenguaje C, con componentes çableados"para correr aplicaciones en TinyOS,para lograr esto, existen las llamadas interfaces, que son canales bidireccionales multifuncionales entre dos componentes, el componente proveedor y el componente usuario. En una interfaz se especifica los comandos que deben ser implementados por el proveedor y los eventos que deben ser implementados por el usuario. Una aplicación NesC consiste en uno o más componentes relacionados por interfaces para formar un programa ejecutable.

RSSI

RSSI es una escala de referencia (1 mW) usada para medir el nivel de potencia de las señales recibidas en un dispositivo en las redes inalámbricas. RSSI indica la intensidad de la señal recibida, no la calidad de la señal. Se puede utilizar para estimar la conectividad de un nodo con otro, idealmente en un espacio libre el nivel de RSSI disminuye con la distancia (inversamente proporcional al cuadrado), pero existe ruido, difracciones, reflexiones y dispersión en el entorno de propagación. Particularmente Tmote Skye utiliza un transductor de radiofrecuencia CC2420.

[dBm]	Interpretación
0	señal ideal, difícil de lograr en la práctica
-40	señal idónea con tasas de transferencia estables. dependiendo
-60	enlace bueno; ajustando TX y basic rates se puede lograr una conexión estable; al 80 %
-70	enlace normal -bajo; es una señal medianamente buena, aunque se pueden sufrir problemas con lluvia y viento
-80	señal mínima aceptable para establecer la conexión; puede ocurrir caídas, que se traducen en corte de comunicación (pérdida de llamada, perdida de datos, mensajes corruptos (ilegibles)

Tabla 2: Interpretación codigos RSSI

Desarrollo

Este proyecto tiene como fin el generar una conexión entre dos *Tmote sky*, enviando mensajes periódicamente desde un *Sender Mote* a un *Receiver Mote*. Luego el *Receiver Mote* al recibir los mensajes podrá leer la calidad del enlace mediante el valor RSSI del mensaje.

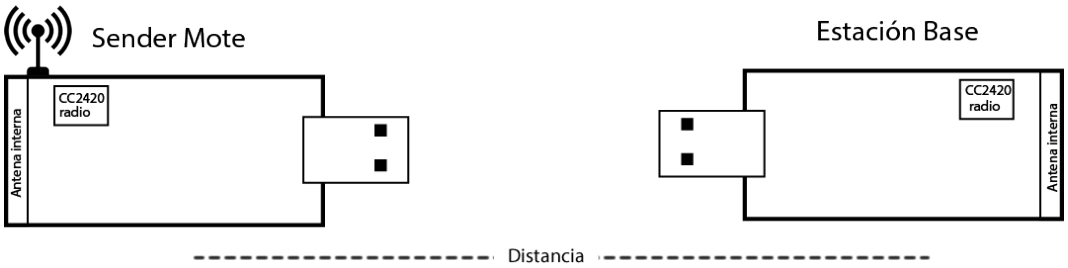


Fig. 3: Modelo de la conexión

Para el proyecto se basó su desarrollo en la aplicación demo RssiDemo ubicada en *apps/tutorials/RssiDemo* proporcionada al instalar *TinyOS*.

La aplicación Demo se puede estructurar de la siguiente manera.

Intercept Base

Intercept Base es una versión modificada de la aplicación de prueba de *TinyOS*, *BaseStation*, que proporciona la interfaz de intercepción, que le permite a una aplicación del usuario inspeccionar y modificar el radio y los mensajes seriales antes de que sean enviados a otro Tmote Sky, y decidir si se retransmitirán o no. Para el desarrollo de este experimento no se incluirá el uso de ningún Tmote Sky con la aplicación *Intercept Base*.

Sending Mote

Sending Mote es la aplicación para ser cargada en un dispositivo Tmote la cual envía el mensaje cuyo Rssi será leído por la aplicación base. Contiene una lógica sencilla que envía periódicamente un mensaje Rssi, como se define a continuación:

```
typedef nx_struct RssiMsg{
    nx_uint16_t rssi;
}RssiMsg;
```

El mensaje Rssi es enviado vacío, y es la base la cual incluye el valor Rssi desplegado por terminal. Esta es la razón por la cual la aplicación *Intercept Base* no es utilizada para el desarrollo del experimento.

Rssi Base

Rssi Base es la aplicación para ser cargada en un dispositivo Tmote conectado al puerto serial para leer efectivamente el Rssi.Contiene algunas macros crípticas para que pueda trabajar correctamente con los chips que utilizan ya sea el radio CC2420, CC1000, RF230, RFA1 o TDA5250. Primero analicemos el evento de envío:

```
event bool RssiMsgIntercept.forward(message_t *msg, void *payload,
uint8_t len) {
    RssiMsg *rssiMsg = (RssiMsg*) payload;
    rssiMsg->rssi = getRssi(msg);

    return TRUE;
}
```

Siempre reenvía los mensajes (siempre devuelve TRUE), pero modifica el *payload* al incluir el Rssi. Analicemos ahora el método *getRssi (message_t *)*:

```
#ifdef __CC2420_H__
  uint16_t getRssi(message_t *msg){
    return (uint16_t) call CC2420Packet.getRssi(msg);
  }
#elif defined(CC1K_RADIO_MSG_H)
    uint16_t getRssi(message_t *msg){
    cc1000_metadata_t *md =(cc1000_metadata_t*) msg->metadata;
    return md->strength_or_preamble;
  }
#elif defined(PLATFORM_IRIS) || defined(PLATFORM_UCMINI)
  uint16_t getRssi(message_t *msg){
    if(call PacketRSSI.isSet(msg))
      return (uint16_t) call PacketRSSI.get(msg);
    else
      return 0xFFFF;
  }
#elif defined(TDA5250_MESSAGE_H)
   uint16_t getRssi(message_t *msg){
       return call Tda5250Packet.getSnr(msg);
   }
#else
  #error Radio chip not supported! This demo currently works only \
         for motes with CC1000, CC2420, RF230, RFA1 or TDA5250 radios.
#endif //__CC2420_H__
```

Este código, para plataformas con radio CC2420, entregan el Rssi desde el método *getRssi (message_t*)* de la interfase del CC2420Packet, entregado por el componente HAL CC2420ActiveMessageC (HAL es referido a un software que provee una capa de abstracción de hardware). Para plataformas con el radio CC1000, el Rssi es extraído del paquete metadata. Para el caso de los radios RF230 y RFA1, se entrega el Rssi desde el metodo *get(msg)* de la interfase PacketRSSI. Por último para plataformas con radio TDA5250, el mensaje Rssi se entrega desde el metodo *getSnr(msg)* de la interfase Tda5250Packet

Aplicación Java

Para poder observar el resultado de la comunicación entre los Tmote, se utiliza una aplicación java, la cual lee los datos de RSSI del puerto serial conectado al Tmote Base, y los muestra por terminal. Al correr la aplicación se debe especificar el puerto a leer, esto se hace agregando la opción *-comm serial@/dev/ttyUSBX:tmote*, donde ttyUSBX (para el caso de linux) corresponde al puerto serial al cual está conectado el Tmote (para Windows se remplaza ttyUSBX por COMX). El comando para correr la aplicaión quedaría como: *java RssiDemo -comm serial@/dev/ttyUSB0:tmote*, donde el puerto a usar en general es el ttyUSB0, si no existese algún otro dispositivo que use comunicación serial. La salida de la aplicación java se puede ver en la Fig 4, en donde se pueden observar los distintos valores observados de Rssi, además de errores en los paquetes recibidos.

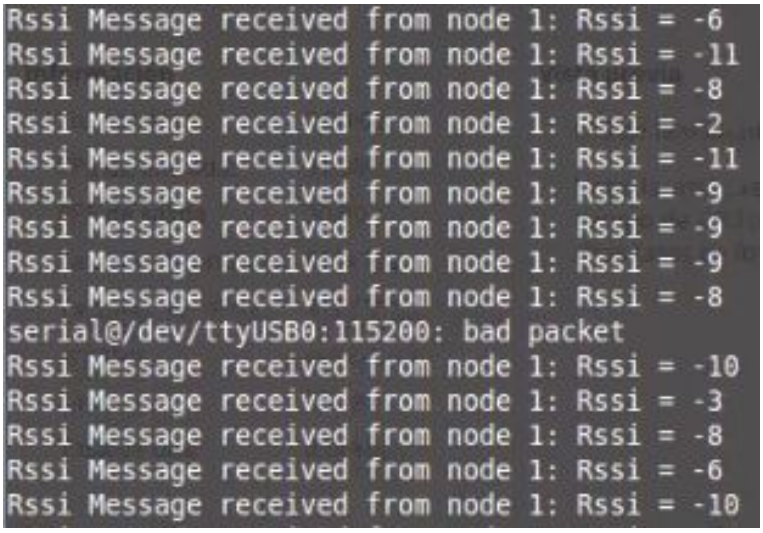


Fig. 4: Salida por terminal de la aplicación java

De esta forma, mientras el dispositivo Tmote con la aplicación *Sending Mote* se esté alejando del dispositivo Tmote con la aplicación base *Rssi Base* conectada al computador, los valores obtenidos de RSSI irán decreciendo, haciéndose mas negativos, siendo esto comprobado experimentalmente.

Resultados

Para realizar el experimento, se efectuó una medición de los datos de Rssi alejando el Tmote con el programa SendingMote del Tmote con el programa RssiBase y luego se acerco volviendo a su separación inicial. De los valores obtenidos de Rssi se obtiene el gráfico mostrado en la Fig 5, en el cual se puede apreciar un comportamiento cuadrático de la intensidad de señal versus la distancia de separación entre ambos Tmotes. De esta manera podemos ratificar que la intensidad de señal es inversamente proporcional al cuadrado de la distancia de separación entre el dispositivo emisor y receptor del mensaje.

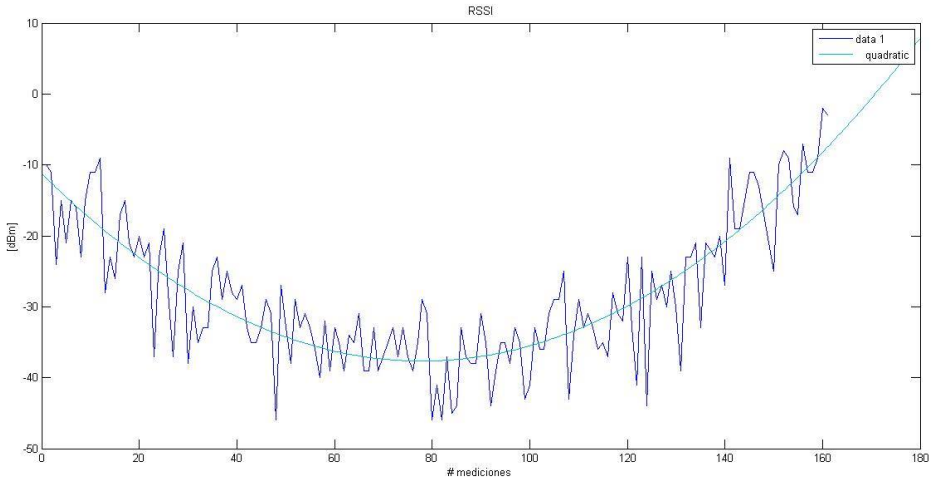


Fig. 5: Gráfico de los valores obtenidos de Rssi de los mensajes recibidos por la estación base

Anexos

Instalación de TinyOS

Para instalar *TinyOS* en *Linux*

1. Agregar el siguiente repositorio de TinyOS a /etc/apt/sources.list Distribuciones soportadas: edgy, feisty, gutsy, hardy, jaunty, karmic, lucid (usada en Ubuntu 14.04)

```
deb http://tinyos.stanford.edu/tinyos/dists/ubuntu
<distribution> main
```

2. Gestionar los paquetes instalables disponibles

```
sudo apt-get update
```

3. Instalación *TinyOS 2.1.2*

```
sudo apt-get install tinyos-2.1.2
```

4. Setear variables de entorno

```
$ cd /opt/tinyos-2.1.2
$ sudo nano tinyos.sh
```

5. Agregar las siguintes lines:

```
#! /usr/bin/env bash
export TOSROOT="/opt/tinyos-2.1.2"
export TOSDIR="/opt/tinyos-2.1.2/tos"
export CLASSPATH="$CLASSPATH:$TOSROOT/
support/sdk/java/tinyos.jar:."
export MAKERULES="/opt/tinyos-2.1.2/
support/make/Makerules"
export PATH="/opt/msp430/bin:/opt/jflashmm:$PATH"
```

6. Modificar bashrc

```
sudo nano ~/.bashrc 1
```

7. Agregar lo siguiente

```
$ sudo tos-install-jni
$ sudo apt-get install g++
```

Referencias

- Tarea Tmote Sky 2010 - ELO323
- Rssi Demo - TinyOs
- Memoria - Universidad Politécnica de Catalunya ffffff22222